Tech., Inst., Cognition and Learning, Vol. 0, pp. 000–000
Reprints available directly from the publisher
Photocopying permitted by license only

Ontology-Based Educational Modelling - Making IMS-LD Visual

GILBERT PAQUETTE*

LICEF Research Center, CICE Research Chair, Télé-Université
100 Sherbrooke St. Ouest, Montréal Québec, Canada, www.licef.ca/gp

This article analyses educational modelling with the IMS-LD specification from the perspective of knowledge-based learning environments. An extension of the MOT visual language covers a wider variety of rule formats to provide control of the flow of activities in a multi-actor process-based scenario. The visual scenarios are executed by an ontology-driven player that constructs from it a web interface enabling interactions between actors and with activities and resources proposed in the scenario. Besides this first use of ontology modelling, domain ontologies, extended by competency statements, serve to reference actors, activities and resource semantically, thus providing a foundation for the design and delivery of knowledge-based learning environments.

Keywords: Educational Modelling, Visual Scenarios, Instructional Engineering, IMS-LD specification, Ontology-Driven Tools and Systems, Knowledge-based Learning Environments.

1. INTRODUCTION

Building pedagogically useful learning scenarios is a difficult task per se. Using standards like IMS-LD and OWL-DL adds new difficulties to the average professor, trainer or instructional designer. These difficulties must be overcome.

With this goal in mind, we have started a research stream with the design of a first instructional design support system called AGD (Paquette, Crevier and Aubin, 1994). From it, an instructional engineering method, MISA (Paquette 2002) and some visual scenario (learning design) modelling tools, MOT and

*Corresponding author: gilbert.paquette@licef.ca

MOT+ (Paquette 1996, 2003) were built. The modelling language is based on a large consensus in education and applied cognitive science (Merrill 1994, Romiszowski 1981, Tennyson and Rasch 1988, and West, Palmer and Wolff 1991).

In 2005, we have elaborated the MOT+LD visual language (Paquette, Léonard, Lundgren-Cayrol, Mihaila and Gareau, 2006) enabling designers to build visual learning scenarios at level A of the specification. The usability of the MOT+LD Visual Editor has been validated by the development of a set of learning designs by designers in four different universities. Around 50 IMS-LD scenarios can be found on the IDLD portal (IDLD, 2007).

In the last three years, we have moved to a another stage with the TELOS Scenario Editor that will be presented here, coupled with a player for IMS-LD scenarios at the three levels of the specification and an Ontology Editor to associate semantic references to scenario entities. Figure 1 summarizes our research and development path.

The new TELOS environment is Ontology-Driven so it embodies a metadata referencing scheme for scenario entities and scenarios as well. The new Scenario Editor covers all levels of the IMS-LD specification using visual symbols and links. The resulting scenario can be exported to an IMS-LD file or executed within the TELOS environment. TELOS includes a player that provides a visual tool, the Task Manager, to enables participants to realize their activities and interact with each others and the environment. This human interface is produced automatically from the scenario visual design. Our actual work aims to reintegrate functionalities from a previous system called Explor@ (Paquette 2001) to facilitate the ontology referencing of scenario components for the design of knowledge-based learning environments.

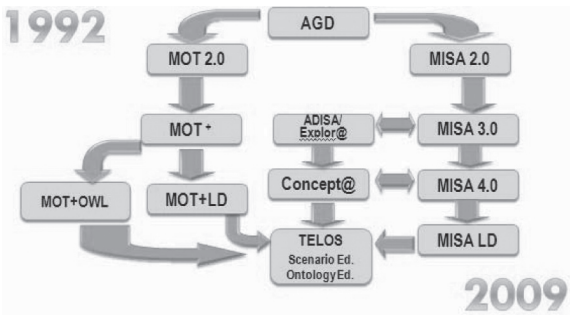


FIGURE 1
Instructional Engineering Research Process.

This article is organized into 4 sections. In the first one, we will present a brief summary of scenario modelling standards and tools, to provide a basis for our own work. In section 2, we will focus on the TELOS ontology-based scenario editor, illustrating its use and presenting its principles of operations. In section 3 and 4, we will develop a proposal for the design of knowledge-based environments through the referencing the resources using domain ontologies and competency statements. In section 5, we will discuss usability and generality issues of this proposal and we will conclude on the progress and limit of this contribution and on the work going on at our research centre on these questions.

2. SCENARIO AND ONTOLOGY STANDARDS AND TOOLS

Basically, reusability means being able to introduce an educational resource or learning object (LO) in new educational contexts or courses that use a variety of interoperable e-learning delivery systems. This goal demands for a standard way of describing those learning objects or educational resources. In the past few years, a vast movement towards international standards for learning resources has been initiated. Duval & Robson (2001) have presented a review of the earlier phases in the evolution of standards and specifications, starting with the Dublin Core metadata initiative in 1995 up to the publication of the Learning Object Metadata (LOM) standard by IEEE in 2002.

Since then a host of other specifications have been published such as the IMS Learning Design specification (IMS-LD) that supersedes the widely used SCORM profile for on-line educational environment. A list of recent specifications have been made available for Question and Test Interoperability (QTI), Learning Tools Interoperability (LTI) and Common Cartridge (CC), the more recent one in the IMS world. Meanwhile, an IEEE working group is aiming at integrating the various competency modeling proposals, including the IMS RDCEO, into a standard way to represent human competencies.[1]

Because on-line learning environments are software systems, distributed on the Internet, other standards, non-specific to education must be considered such as the Business Process Modelling Notation (BPMN 2006) published by the Object Management Group (OMG), or the Ontology Web Language (OWL) published by the World Wide Web Consortium (W3C 2004).

In this section we will focus on standards and tools for learning design and workflow modelling where we will consider the IMS-LD specification and the

[1] All available at http://www.imsglobal.org/specifications.html

BPMN standard, in order to set the basis for the construction of the TELOS Scenario Editor.

2.1 eLearning Standards and IMS-Learning Design

High quality learning objects are necessary but not sufficient to produce a high quality course or unit of learning. When, how, for what and by whom will those LOs be used? The fast evolution of learning technologies has multiplied the number of decisions one must take to create a distributed learning system. While it is true that a majority of the first Web-based applications have been mostly used to distribute information, more and more educators have become aware of the need to go beyond simple uses of information and communication technologies. This context has generated a much-needed interest for pedagogical methods and, more generally, for the field of Instructional Design (Wiley 2002).

The term "Educational Modeling Language (EML)" was first introduced in 1998 by researchers at the Open University of the Netherlands (OUNL), as a response to Instructional Design and pedagogical concerns towards standardization and interoperability needs. The work on Educational Modelling Languages (Koper 2001), and the subsequent publication of the IMS Learning Design Specification (IMS-LD 2003), is the most important initiative to date, to integrate Instructional Design preoccupations into the international standards movement by describing a formal way to represent the structure of a Unit of Learning and the concept of a pedagogical method.

The IMS-LD specification was approved in February 2003. In the following year, it had been downloaded 10,444 times[2] and had trigger a blurring R&D activity. The specification consists of three documents available from the IMS web site:

- *IMS Learning Design Information Model,* describing the conceptual model and data structures, as well as the behavioural model and runtime behaviour;
- *IMS Learning Design Information Binding,* providing detailed information on each of the elements in the specification's XML binding;
- *IMS Learning Design Best Practice Guide,* describing how to implement an IMS-LD specification and providing both examples of structured learning scenario narratives and corresponding XML documents. It also provides an implementer's guide.

As shown on figure 2, the IMS-LD information model specifies three embedded levels of implementation and compliance, each with its separate XML

[2] Personal communication from Lisa Mattson from IMS in March, 2004.

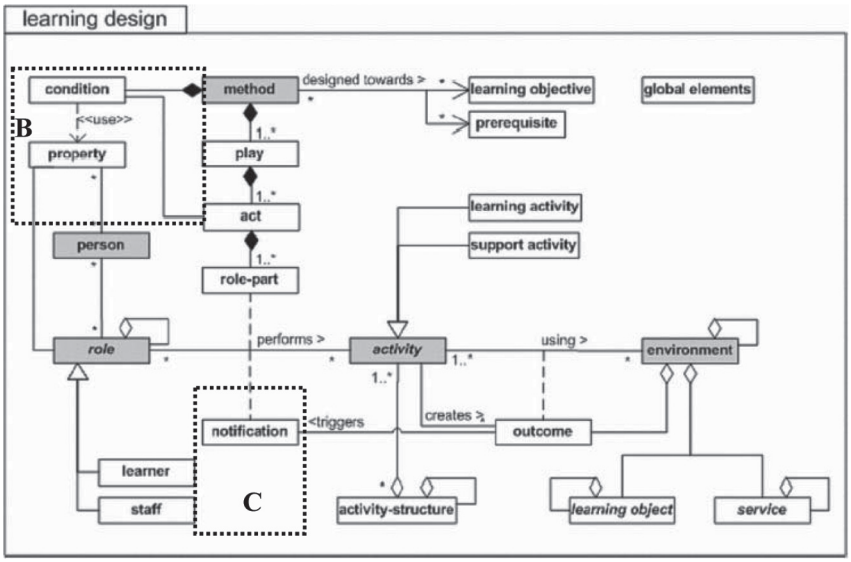


FIGURE 2
The IMS Learning Design Conceptual Model.

schema. Level A contains all core elements, roles, activities and environments (learning objects and services). Level B adds to Level A properties and conditions for user modelling, assistance and personalization. Level C adds notifications between actors to support collaboration and tutoring.

The Level A model involves three entities: roles, activities and environments.

- *Roles*, such as learner of staff (facilitator, professor, tutor) are played by persons described by their properties;
- *Activities*, performed by roles are organized in a tree structure called a method, decomposed into alternative plays, themselves decomposed into a sequence of acts, each act being decomposed into activity structures, which contain other activity structures down to terminal learning or support activities;
- *Environments* group all kinds of learning objects (or resources) and services, as well as outcomes produced by roles while performing the activities.

When activating a unit of learning, the method element is central. This element and its sub-elements describe the learning process and control the behaviour of the unit of learning as a whole, coordinating the activities of the players in their various roles and their use of learning resources.

IMS-LD embeds and generalizes other IMS specifications such as MD (metadata), SS (simple sequencing), CP (content packaging), RDCEO (learning objectives and prerequisites), QTI (questionnaires and tests), LIP (learner profile) and others.

SCORM, the Sharable Content Object Reusable Model proposed by the ADL Technical Team (2004), while sometimes seen as opposed to IMS-LD, can be seen as a specialization to single-user activity structures. Although SCORM integrates some of the IMS specifications embedded in LD such as MD/LOM, SS and CP, a SCORM delivery system cannot deliver a complete IMS-Learning Design. But conversely, LD expands SCORM specifications in many ways:

- LD describes methods as multi-actor workflow processes, instead of single actor activity tree;
- LD can provide alternative plays adapted to different target populations and delivery modes;
- LD integrates the description of collaboration services;
- LD integrates (at Level B and C) same user modelling and notification between the different actors;
- Finally and most important, LD favours instructional strategies like collaborative learning, problem solving, project-based learning, communities of practices, and multi-facilitators support as found in more advanced learning strategies.

The *learning design* is an abstract model of a unit-of-learning that can be instantiated before a delivery session or during delivery. Instantiation means that a concrete person, learning object or service is put at the place of a role or an environment in the LD model. For example, the participants in a forum are variables in the model. They can be specified by concrete persons (giving their email) when a delivery starts or at runtime (during delivery). In the same way, a concrete document can be embedded in the design or kept open as a variable in the LD model. In this case its address will be specified later at instantiation time, during the delivery.

In principle, a learning design built in the IMS-LD format can be reused on any machine properly equipped. This is the goal of the specification, to bridge the gap between the process of designing a course and that of delivering it.

Concretely, a unit of learning is described as an XML file called a *content package*. These content packages contain all the necessary information on a unit of learning, in an XML format that can be read by any compliant delivery system or platform.

Figure 3 shows the structure of an IMS-LD package, the central part of which is the *manifest*.

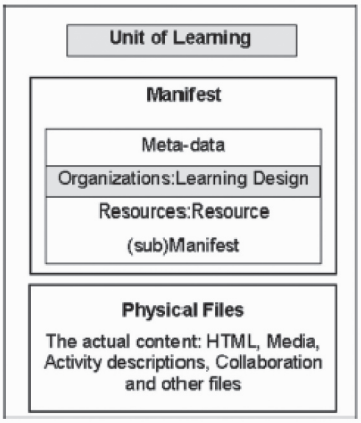


FIGURE 3
Structure of an IMS-LD XML package.

- The manifest contains the metadata of the UoL, in particular a fixed, pre-defined name to help find it, and possibly other properties.
- The organization part is the learning design structure described above, with role-parts associated to environments grouping the resources.
- The resources part list all the resources (roles, activities, learning objects, services, prerequisite and learning objectives) included in the design with their addresses if they are specified at design time. It can also contain sub-manifests if other UoL are embedded in the design. The physical files corresponding to the resources can be included or not in the content package.

2.2 IMS-LD Edition Tools

With regard to the tool set, form-based tools like RELOAD, although an improvement from previously used XML editors, impose too many constraints on the design process. For example, learning objects need to be first declared in one form, and then integrated to environments in another form, and still linked to activities in another. At level B and C, properties and conditions are all defined in a list associated to the method element, thus blurring the understanding of the locus of their action.

Visual representation techniques and tools can free instructional designers from these constraints. Although well suited for software engineering purposes, UML graphs and diagrams, as proposed by the IMS Learning Design Best Practice and Implementation Guide (IMS-LD 2003), are not suited for instructional design, except maybe in very simple cases. Complex Learning scenarios, espe-

cially those involving many actors, with conditions and notifications, are not easily represented using UML use cases and activity diagrams. Moreover, it is important that all the IMS-LD components can be integrated using only one graphical model. This can greatly reduce the learning curve for designers.

There exists more user-friendly instructional visual design software like LAMS (Dalziel 2005), or our own MOT editor supporting the MISA method. Although useful for an ideation phase, these tools are too informal and not powerful enough to produce interoperable and executable IMS-LD files. This has led us to build a preliminary version of the MOT+LD specialized editor that produced level A IMS-LD files.

We have also studied another emerging standard in the field of business workflows. The Business Process Modeling Notation (BPMN 2006) places a large emphasis on the flow of control in a process, but is weak on modeling the resources and the knowledge used or produced during the learning process. A comparative analysis has been made between business workflows and IMS-LD learning designs (Marino, Casallas, Villalobos, Correal and Contamines 2006). It has lead to the identification of 21 control situations for workflows encountered in the software engineering literature (Correal and Marino, 2007) that subsume the properties and conditions in IMS-LD, level B and C. Finally, it was decided to combine both notations using our own MOT graphic language, in order to build a user-friendly, yet powerful new scenario editor that will be presented in the next section. To comply with the IMS-LD specification, we have built an export module to IMS-LD, and also an import module from IMS-LD. In this way, we can keep the inherent complexities of IMS-LD hidden from the user, while keeping the interoperability of the new software with other IMS-LD compliant editors or players.

3. VISUAL SCENARIOS DESIGN AND ONTOLOGY-DRIVEN EXECUTION

We now introduce the main tools of the TELOS system[3] from a user's viewpoint. We will present the graphic forms and links that are used in the new TELOS Scenario Editor to underline how we cover level B and C of the IMS LD specification. In section 4, we will present the TELOS Ontology Editor with more detail and discuss its use to provide knowledge and competency referencing of resources within a scenario.

[3] This system has been produced within the LORNET research network (www.lornet.org) led by the author and financed by the NSERC Research Council of Canada.

3.1 The Global TELOS Interface

TELOS is an assembly system for on-line environments based on the aggregation of actors, activities, resources and conditions in a scenario. The TELOS user interface is available to all kinds of actors through a Web browser. As shown in figure 4, there are four main tools open: the *Resource Manager*, the *Scenario Editor*, the *Ontology Editor* and the *Task Manager*.

The *Resource Manager* gives access to all the resources known by TELOS, whether they are documents, tools, operations, functions or scenarios, actors and ontologies. The resource folders on the left side of the corresponding window are not just folders; they are classes of the TELOS technical ontology that have been built and can be modified with the *Ontology Editor*.

This editor helps also to build domain ontologies that describe the knowledge embodied in other resources; these are also stored in the Resource Editor classes of the TELOS ontology within the folder "SemanticResources". The action of putting a resource in a class folder means that it is declared as an instance of this

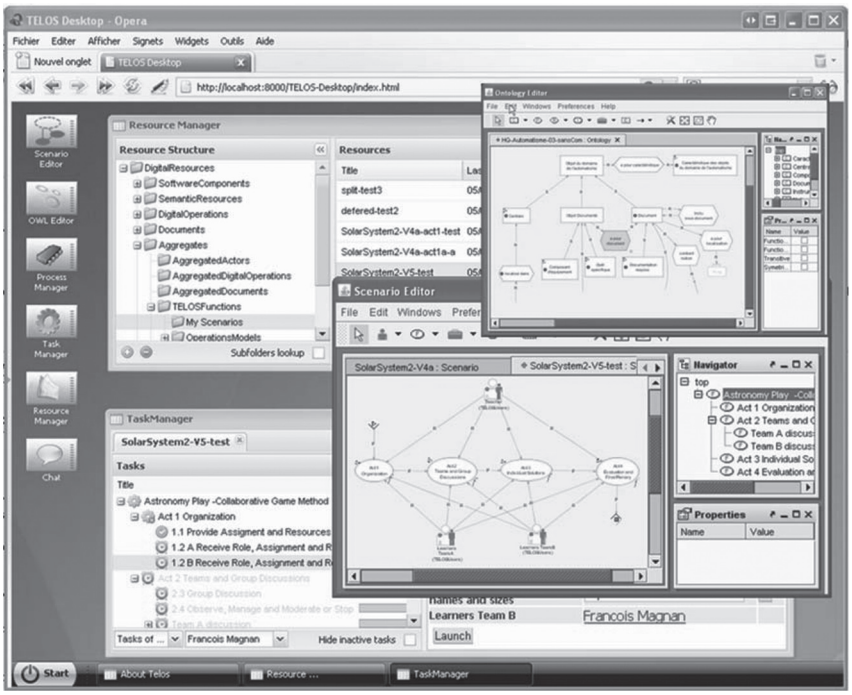


FIGURE 4
A view of the TELOS interface and the main TELOS tools.

class. This action semantically references that resource in terms of the class properties defined in the ontology, which provides its execution metadata. Such semantic references will inform the system, for example, that the resource, depending on its type, can be displayed either as document, launch as a software component, be used to notify a person or as a metadata source.

The *Scenario Editor* will be described in the next paragraph. Basically, it is a resource aggregator. It helps to orchestrate interactions between actors in activities where they use, process and produce other resources. A scenario player acts behind the scene to execute the scenario and make the appropriate processing of the resources as the scenario unfolds at runtime.

The *Task Manager* is the runtime user interface of the scenario player provided to the participants in the scenario. It is a multi-actor interface with both a general tree view and a local visual view on an actor's and other actor's activities. It provides to its users a form of telepresence that evolves as the scenario is undergoing.

3.2 TELOS Visual Scenario Editor

The Visual Scenario Editor is the central piece of the TELOS architecture (Paquette, Rosca, Mihaila and Masmoudi, 2005; Magnan and Paquette 2006). Extending the MOT visual language, the Scenario Editor uses four kinds of objects (Actor, Function, Resource, Condition) with subtypes related to the TELOS technical ontology that drives the system. These symbols are not necessarily in one to one correspondence with IMS-LD terms. For example, the Function symbol can represent Methods, Plays, Acts, as well as Activity Structures. Differentiations between such terms can be made either in the property sheet of the object or be deduced by the Export-to-LD parser when it translate automatically a graph into the IMS-LD XML file.

These symbols are shown on figure 5. MOT Concept symbols serve to represent all kinds of *Resources*: documents, tools, semantic resources, environments, resource-actors, resource-activities and datatypes. MOT Procedure symbols (ovals) represent *Functions* (groups of resources that together achieve a function process). Functions can be decomposed into other functions at any depth level down to activities enacted by humans, or operations performed automatically by the system. Finally, MOT Principles serve to represent actors as well as conditions. The *Actors'* symbols represent users, groups, roles or software agents, seen as control objects that enact the activities using and producing resources as planned by the scenario model. The *Condition* symbols (hexagons) represent control element inserted within the basic flow to decide on the following activities or operations. The diamond shapes defines the start and end point for activities.

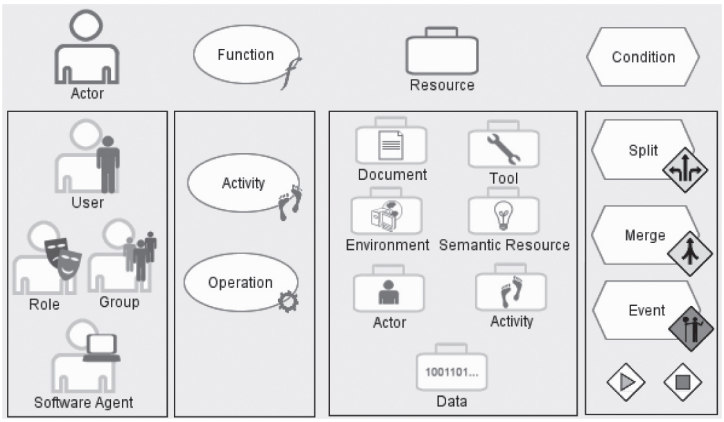


FIGURE 5
TELOS Scenario Editor (TSE) Visual Symbols.

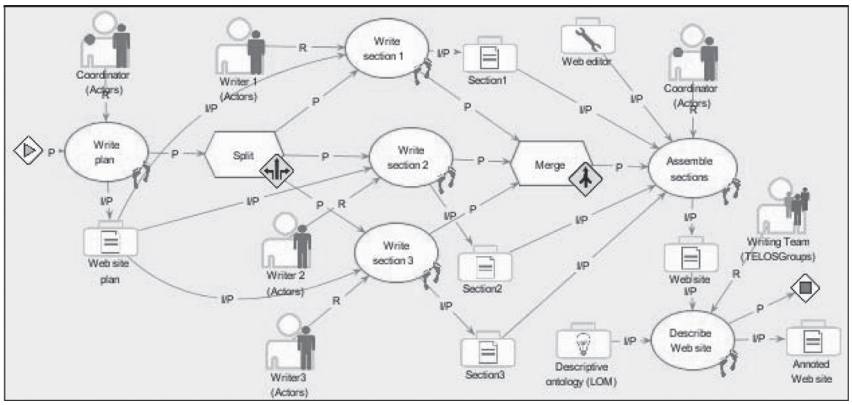


FIGURE 6
A simple scenario model.

On figure 6, we see a scenario that combines some of these symbols. A coordinator writes the plan of a document in the first activity. Then three activities are performed in parallel by different writers to produce the three sections. When these are all terminated, a Web site grouping the different parts is built by the coordinator using a Web editor, and this site is annotated by the group to describe it using metadata and/or ontologies, in order to produce the annotated Web site. This example shows a split condition after the first activity. Later on, the flow from the activities merges through the merge condition before the next activity takes con-

trol. According to the merge condition properties, the "Assemble sections" activity will wait for all the incoming flows to terminate before it is executed.

In the Scenario Editor, there is a combination of a control flow and a data flow. The control flow is modeled using the MOT basic P and R links. P links (precedence) are used for the basic line of execution to indicate the sequencing between Functions, Activities and Operations. R links (regulation) identify the source of an event (from a user or from the system) that triggers a condition that will alter the basic flow of control. MOT I/P links (input/product) serve to model the data flow, either from resources to activities where they are consulted, used or processed or from activities to outcome resources.

In TELOS, the Scenario Editor can be used at all levels of the system. It enables engineers to combine software components into larger ones, technologists to build platform workflows (instructional methods) for designers, and finally designers to build courses or work scenarios. Examples of these are presented in (Paquette and Magnan, 2008).

3.3 Ontology-Driven Scenarios

The Scenario Editor is ontology-driven in the sense that the execution of a scenario depends on the association of its objects to classes of the TELOS technical ontology described in the OWL-DL format (W3C 2004). All the pieces of TELOS must fit in this ontology, which is the logical blueprint of the system. The TELOS technical ontology is not only a conceptual representation of the architecture of TELOS with all its main components. It is an internal way to guide the execution of TELOS. Here we have used a software development strategy in which core functionalities are programmed in ontologies and in the queries we send to the inference engine processing this ontology in order to make deductions that produce the system's services and behaviour. In the above sense TELOS is an Ontology Driven Architecture (Tetlow et al 2001). This approach combines very well with the concepts of Service Oriented Architectures (Wilson, Blinco and Rehak 2004), which is also a key orientation of the system. These architectural principles are explained in (Paquette and Magnan 2008).

In the example in figure 7, we see a selected document in the back window used in the activity labelled "team A discussion". On the property sheet at the right lower corner, an execution semantic has to be chosen if we want the scenario to be played. Such a selection corresponds to telling the system what are the properties of this graphic object according to the TELOS technical ontology. Selecting that property opens the window shown at the centre of the figure. This window is a view of the TELOS ontology where we can select a class of resources (to be instantiated by the actors at run time) or a specific instance to be displayed

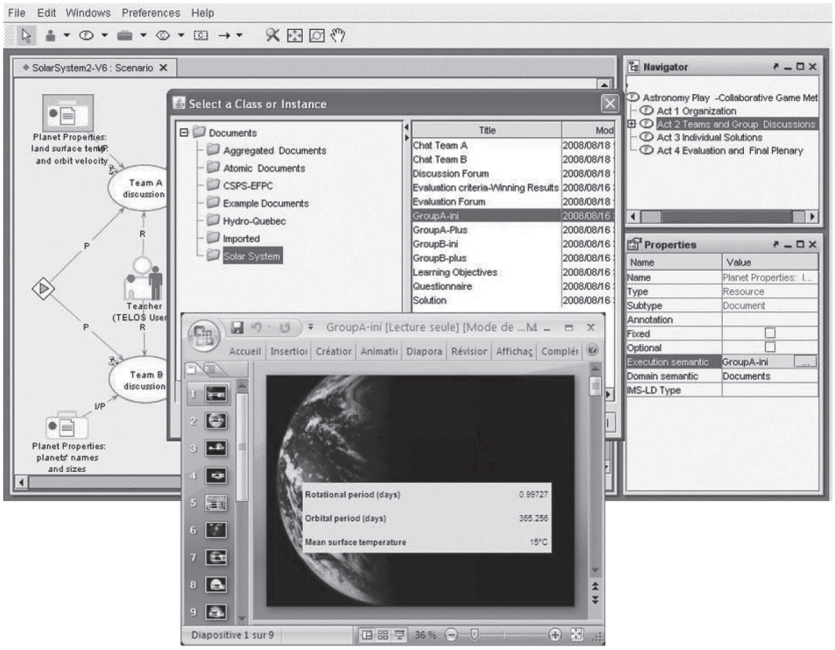


FIGURE 7
A view of a scenario and the semantic referencing of a resource.

at run time. In this example, the resource is an instance, a specific Powerpoint presentation on planet properties labeled "GroupA-ini", also shown on the figure. By this association, we tell TELOS that this is not an actor, an activity, an operation or a condition, but a document that the system should open at run time when the time comes.

3.4 A Scenario Case Study

In figure 8, we display a model of an act of a solar system astronomy unit-of-learning, opened in the TELOS Scenario Editor. On the left side, we see the upper model of Act 2, where the flow splits into team discussions. The execution waits for both discussions to end and then moves to the group discussion forum where members from both learners' teams and the teacher will all join in.

In the right hand part of figure 8, we see a sub-model for the team A discussion. It starts with opening the chat service for team A. Then, the control splits between the learning activity 2.1.A, where team A learners discuss documents on planet properties, and the support activity 2.2.A performed by the teacher where

he observes the team A discussion. The teacher's part is highlighted on the figure. After a certain time in activity 2.2.A, the teacher can either stop the discussion or provide additional information (Clue A) to help the learner solve the problem. The learners can also decide to stop, either before or after they have received this additional information. A similar pattern rules the discussion for team B, with the same teacher acting as a facilitator for both teams, each with a different set of planet properties as additional information.

The conditions shown on figure 8 are rules expressing the equivalent of IMS-LD level B properties. The decision "Need Clues or Stop Team A?" depends on its input data and the value "true" or "false" that a teacher action will produce in activity 2.2.A for these input variables of the condition. If the value "Stop team A discussion" is true, then the flow of control goes to the end symbol, after which the flow goes up to the main act 2 model to the Join condition. If the value of "Clues A needed" is true, the flow will proceed to the teacher's activity where he selects a document named "Clue A" to be provided to the learners. If both input variables are false, the flow will come back to activity 2.2.A where the teacher

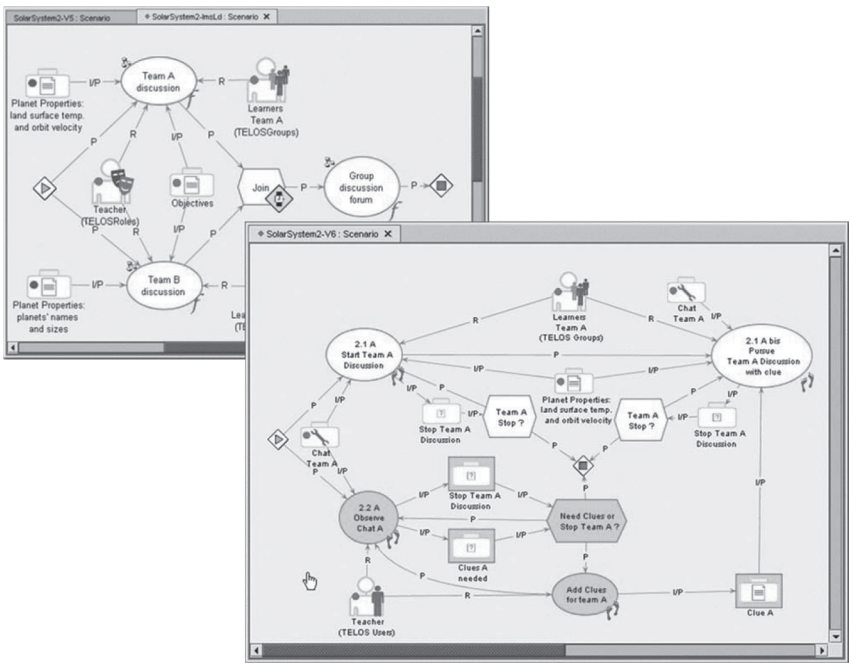


FIGURE 8
A view of a scenario for act 2 of the Planet Game unit-of-learning.

will go on observing the chat. In the task manager, a Web interface is provided to the Teacher actor to enter a value in the data objects "Stop Team A Discussion" and "Clues A needed". Depending on these values, the condition "Need Clues or Stop Team A" will be executed to orient the flow of activities.

Figure 9 shows three views of the task manager of the same run of the Planet Game scenario. The one on the left is the teacher view selected at the bottom of this window. Since the teacher takes part in all the activities, he will see a tree view of all the activities.

Some of the activities are Xed because they have not been completed. The reason is that in the team A discussion, the learners have put an end to the chat without having to use any clue. We see this on the second window that shows what team A members see of the scenario. On the contrary, in the team B discussion, the team members have continued the discussion with a Clue provided by the teacher and later on, the teacher has put an end to the team B discussion, thus completing his own activities. The last window on the figure shows the view provided to team B members.

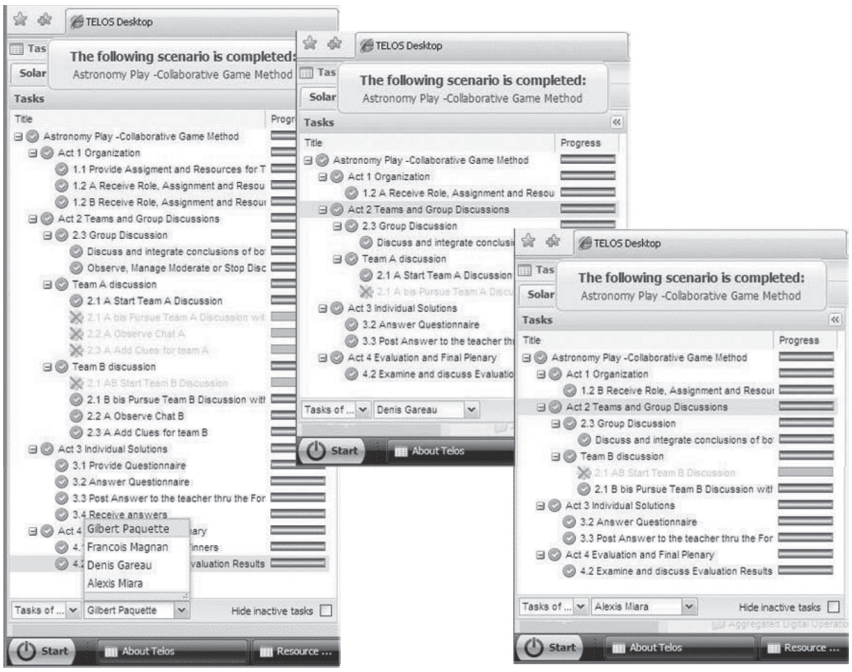


FIGURE 9
The multi-view aspect of the Task Manager.

Another feature of the task editor helps any user see where the others are in the scenario during its execution. Some of the documents may be made not visible to others.

At run time, we found out that the Task Manager interface provided adequate observation facilities of the learners' performance. For example in Act 2, the teacher could look at the messages in the chats of each team and decide what to do, stop the discussion to move to the general forum activity, or add a clue if a team seemed to have some difficulties with the content. He could also use the communication services at any time to send messages to the learners and interact with them to monitor their progress.

The scenario we have built for this case study could be re-used or adapted in many ways. One way is to keep the structure of the scenario (or pattern) but replace the resources by others for a totally different subject-matter area such as literature or management. We could also adapt the structure in the TELOS Scenario Editor to add more teams for a larger class; this would require copying some graphs with slight modifications. We have also decomposed the scenario into the four acts and stored them in the "Aggregates" section of the Resource Manager for future retrieval and recombination in a different order or with other "nuggets" extracted from other scenarios.

4. DESIGNING KNOWLEDGE-BASED LEARNING ENVIRONMENTS

Without any representation of the knowledge to be processed in an eLearning environment, a delivery system will be unable to help its users according to their present and expected state of knowledge and competency acquisition.

In (Paquette and Marino 2005) we have discussed briefly the strengths and weaknesses of the IMS-LD educational modeling specification. Despite its high value, IMS-LD is weak on knowledge representation of the actors, activities and resources that compose a learning design. Actually, in IMS-LD, the only way to describe the knowledge in the activities or in the resources is to assign optional educational objectives and prerequisites to the unit of learning as a whole and/or to all or some of the learning and support activities. Objectives and prerequisites correspond to entry and target competencies. They are essentially unstructured pieces of text composed according to the IMS RDCEO specification (IMS-RDCEO 2002).

Unstructured texts are difficult to compare. Consistency checking between different levels of the LD structure cannot be supported computationally. Even at the same level of a learning design, for example within an act, no relations exist

between the knowledge processed in learning activities and the knowledge present in input or outcome resources, or the actors' knowledge and competencies. In fact, in IMS-LD the knowledge represented in learning resources is not described at all, and the actor's knowledge and competencies are only indirectly defined by their participation in learning units or activities, as long as educational objectives have been associated to the activities.

Earlier, we have briefly presented the use of the TELOS Ontology Editor in its role to build and maintain the technical ontology that drives the execution of TELOS scenarios. We will now present it in more detail and illustrate its use for the semantic referencing of actors, activities and resources using knowledge and competencies in a particular domain.

4.1 Knowledge Representation using the TELOS Ontology Editor

Knowledge in a subject domain can be represented in many ways: taxonomies, thesauri, topic maps, conceptual graphs and ontologies. We have selected to use OWL-DL ontologies for TELOS applications for a number or reasons. It is one of the three ontology Web languages that are part of the growing stack of World Wide Web consortium recommendations for the Semantic Web, the next generation of the Internet. Of these three languages, OWL-DL has a wide expressivity and its foundation in descriptive logic (Baader et al 2003) guarantees its computational completeness and decidability.

OWL-DL provides a precise XML schema but no graphic representation per se. Some ontology editors like PROTÉGÉ, provide interesting graphical views of an ontology, but the main operations are essentially form-based. Our goal here is to provide a complete formal graphic representation of OWL-DL that could combine the virtues of visual editing and still yield a standard format that can be processed by OWL-DL compliant software.

In the context of the MOT representation system, ontologies, in particular OWL-DL constructs, correspond to a category of MOT models called theories. Ontologies can thus theoretically be modeled graphically using the MOT syntax. While doing this, we found out that although the MOT primitive objects and links were sufficient to represent ontologies expressed in OWL-DL, the graphs would become cumbersome unless new symbols were added. We have thus specialized the MOT language and its graphic editor to MOT+OWL (Paquette 2008).

Three types of MOT entities, shown in figure 10, are needed to represent OWL-DL models. Concepts represent *classes* (rectangles), principles (hexagons) represent *properties*, facts (corner-cut rectangles) represent *individuals*. On these graphic entities, we add little icons or special links between them to represent DL constructors or axioms. For example, the group on the upper right corner enables

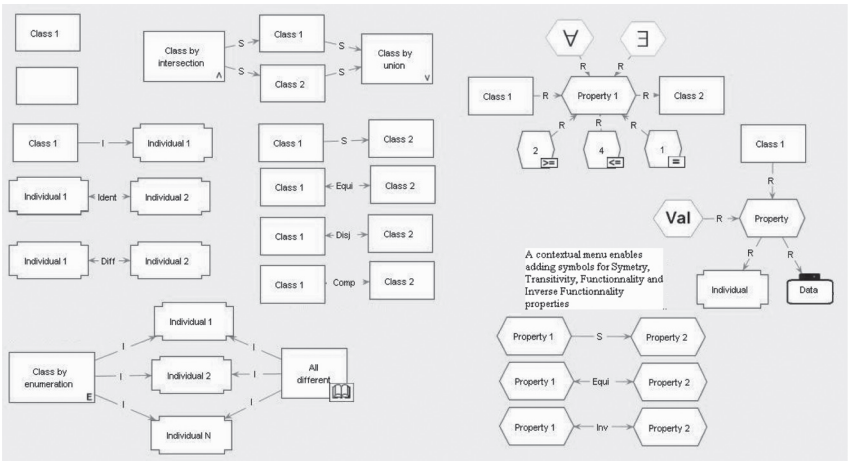


FIGURE 10
MOT+OWL Set of primitive symbols.

the ontologist to declare universal, existential or cardinality restriction on a property "Property 1", in order to define the domain class "Class 1". All these symbols are translated to an XML file that is compliant with the standard OWL-DL schema.

Using these visual symbols, it becomes possible to built more or less sophisticated ontologies like the one on figure 11, a description of an African ecology system.

This example shows that an ontology model can express very precise definitions of classes and objects in a subject-matter domain.

Note for example the following assertions from the figure, expressed in natural language:

- *giraffes* are kinds (S link) of *herbivores*, which are kinds of *animals*; so are *lions* and *carnivores*;
- *herbivores* are *animals* that eat something (anonymous class) that are *plants* or (union label) something (again anonymous class) that are parts of plants;
- *giraffes* only (universal quantifier) eat *leaves*, which are part of *branches*, themselves part of *trees*;
- *is-part-of* is a transitive property (T label); *eats* is an inverse property of *eaten-by*;

Note also an inconsistency in the graph where carnivores are declared as eating only animals, but also that tasty-plants are eaten by carnivores. Putting

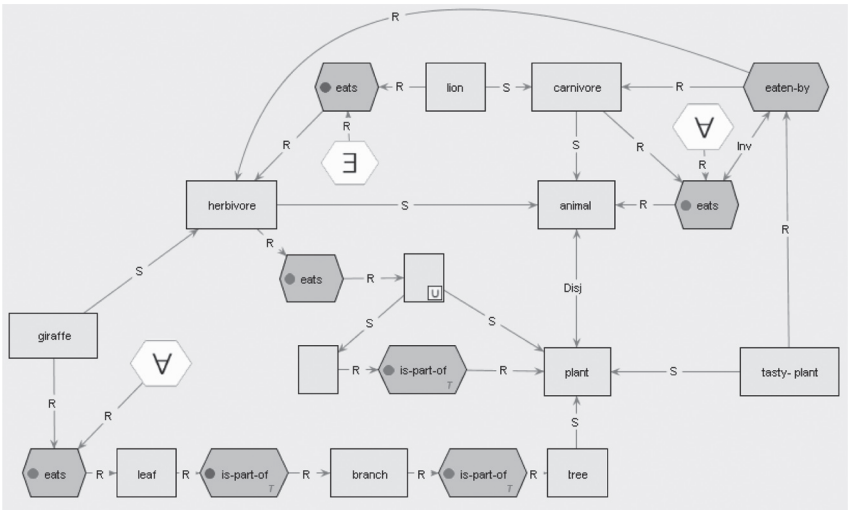


FIGURE 11
A wild-life ontology.

an inference engine at work on this ontology would normally reveal such inconsistencies, helping the ontologist to build consistent theories of a knowledge domain.

4.2 Semantic Annotation of Learning Design Components

We now need a representation of knowledge and competencies involved in activities, resources and actor's roles. This can be done using domain ontologies such as the one presented above to which we add competency attachments.

Figure 12 presents another example where the learning scenario of figure 7 has been put side by side with part of a MOT+OWL domain ontology for the solar system. We define *ontology referencing* as simply a mapping from the ontology to the learning design that associates knowledge elements to components of the learning design. In the figure, we see that data on the orbital period of planets in the solar system has been associated to a learning object in the design, which in this case is a powerpoint presenting this data to team B. This resource is an input to learning activity "team B discussion". But it is not the only input to this activity.

There is also a chat between team B members that will bring additional information to each participant. As a result, the sub-model of the ontology associated to this activity should logically correspond to the union of the sub-models of all input resources to the activity. Finally, the figure shows that most of the ontology

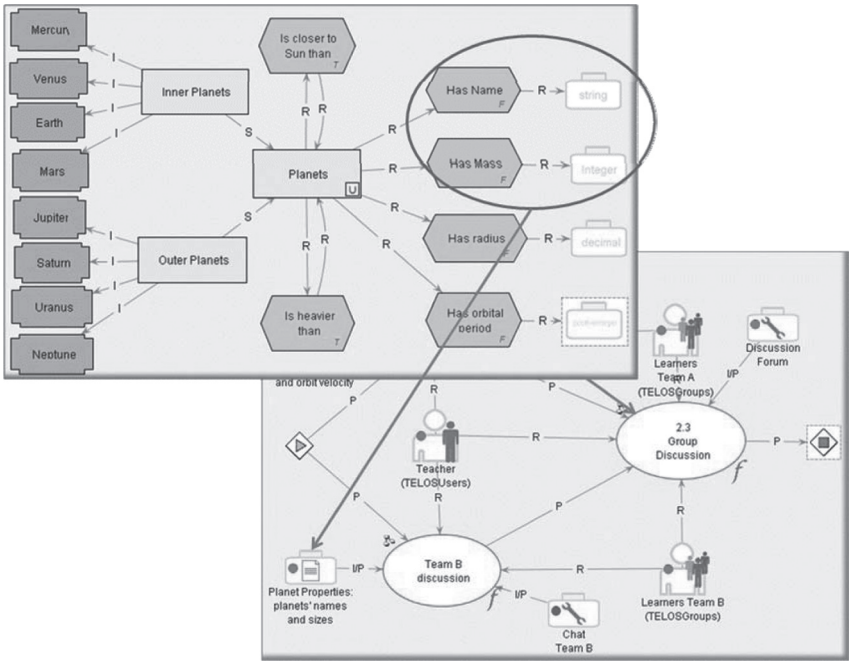


FIGURE 12
An Example of Ontology Referencing of the Resources in a learning design.

model should be the subject of the whole group discussion, since there is another team, team A, that has information on the other properties of the planets.

This example shows how ontology referencing can help guide the construction of learning designs or evaluate their coherence. By associating the right amount of knowledge to the different resources and activities, a designer can build a coherent design that will trigger collaboration between learners, help a trainer decide on its intervention, guide the actions of an intelligent tutoring system, and, in general support the assessment (informal or formal) and the evolution of the learners' competencies.

4.3 From Knowledge to Competency Referencing

Associating knowledge from an ontology to components of a learning design is essential but not sufficient. To say that a person possesses, that an activity involves or that a document contains some knowledge is not enough. Is that person able to give examples of that knowledge? Is she able to apply it, to add elements to it, or to evaluate it according to some criteria? Is a document aiming

simply at presenting the knowledge or is it providing a thorough synthesis of the knowledge ?

If we use only knowledge annotations from an ontology without stating the mastery level of that knowledge, we limit ourselves to a coarse granulation of sense and, as a consequence, to weak support services to the learner. In other words, we need measures of knowledge mastery, a weighted ability defined on that knowledge that corresponds to the concept of competency related to knowledge.

The Explor@-2 delivery system (Paquette 2001) has been based from its inception on two structures, the instructional structure, corresponding to the learning design, and the knowledge/competency structure, corresponding to a simplified domain ontology (in fact a tree of concepts) augmented with attachments from a competency/performance ontology. Here, we aim to expand this association, replacing the tree of concepts by a more powerful domain ontology model and adding to its elements competency attachments similarly based on a competency ontology.

This competency ontology has been presented in much detail in (Paquette 2007). Our definition of a *competency* is founded on the relation between specific knowledge in an application domain and generic skills. *Competencies are statements that someone, and more generally some resource, can demonstrate the application of a generic skill to some knowledge, at a certain degree of performance.*

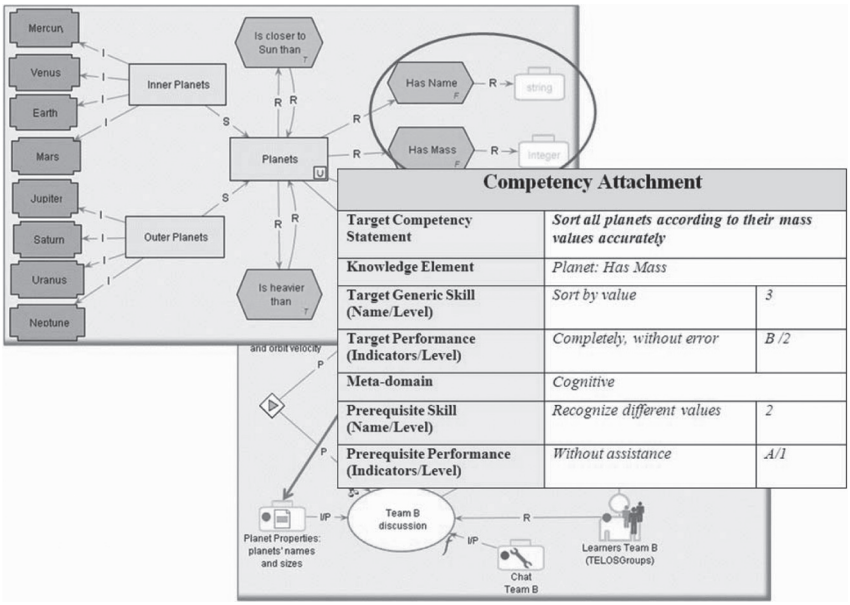
This definition can be seen as an operationalization of the following broad definition*:* Competencies can be conceptualized as complex ability constructs that are closely related to performance in real-life situations. (Harig, Klieme and Leutner, 2008). A thorough discussion of competency, generic skills, and performance exceeds the scope of this text so the reader is referred to previous publications (Paquette 1999, 2002, 2003) for the discussion of a generic skill taxonomy and its relation to knowledge and performance.

The taxonomy of skills is central. It combines elements of an artificial intelligence taxonomy (Pitrat 1990), a software engineering taxonomy (Breuker and Van de Velde, 1994; Scheiber et al. 1993) and two educational taxonomies (Bloom 1975; Romiszowski 1981). The generic skills' taxonomy is ordered by layers from general to more specialized skills. The second layer has ten classes of generic skills ordered by the property "is more complex than". This gives us 10 levels ordered from 1-Pay-attention to 10-Self-manage. The generic skills also have another data-type property, "has meta-domain" that can have as value "cognitive", "affective", "social" or "psycho-motor", as well as any combination of these values.

A generic skill can be made more precise and operational by adding combination of performance indicators such as frequency, scope, autonomy, complexity and/or context of the use. For example, a competency like "diagnose the source of malfunction of a car engine" could be made more precise by adding at the end performance indicators like "in all cases" or "in the majority of cases" (frequency), "for part of the causes" or "for all causes" (scope), "without help" or "with little assistance" (autonomy), "for high complexity engines" (complexity), or "in unfamiliar cases" (context of use). A combination of these value can be simplifies into four classes such as: "awareness", "familiarity", "productivity" or "expertise", or simply by corresponding numbers 2,4,6 and 8 on a 1–10 performance scale.

These categories or levels can be detected by assessment results collected during run-time to provide actual competencies, or they can be calculated from the other indicators at design time to define target competencies. The gap between target and actual competency for a learner is significant. It can help select appropriate resources for a learner.

The example in figure 13 presents one way to associate competency attachment to the ontology presented on figure 12. Here the competency attachment to



Competency Attachment

Target Competency Statement	***Sort all planets according to their mass values accurately***	
Knowledge Element	*Planet: Has Mass*	
Target Generic Skill (Name/Level)	*Sort by value*	*3*
Target Performance (Indicators/Level)	*Completely, without error*	*B /2*
Meta-domain	*Cognitive*	
Prerequisite Skill (Name/Level)	*Recognize different values*	*2*
Prerequisite Performance (Indicators/Level)	*Without assistance*	*A/1*

FIGURE 13
A Knowledge and Competency Editor.

the *Has Mass* property is "*Sort all planets according to their mass values accurately"*. Competency statements like these are texts that are used mainly for display purposes but they have here a precise interpretation as knowledge/(skill + performance) couples. In this statement, the knowledge element provided by the domain ontology is *Planet: Has Mass*. The target generic skill that learners will have to apply to this knowledge is *Sort by value*, which correspond to level 3 skills (Interpret) in the generic skills' taxonomy. The performance level for the application of the skill is set at *B-Familiarity* or level 4 of the performance scale, and the meta-domain is *cognitive*.

The generic skills scale and the performance levels are two ordered sets of values that enable comparison between competency statements for the same knowledge element. In figure 13, we note that the competency attachment contains also a statement of an entry competency, applied to the same knowledge element. The learners must be able to recognize different values (skill level 2: Identify) without assistance (performance level: 1-awareness). Combining the two level values, both for the target and the entry competency, we evaluate that learner will have to increase its competency towards planet masses, from a 2.1 level to a 4.2.

When the ontology elements Planet: Has Mass will be associated in the learning design to a document like Planet Properties: planet names and sizes, it will bring with it the competency attachments if there are any. By this association between the two models, all the actors, activities and resources can be annotated with knowledge and competencies. This example shows one way to extend a knowledge referencing to the competency referencing of resources in a learning design. The evolution of a learner on a competence scale for a certain knowledge element represents a learning progress of mastery of the knowledge therefore, it should be managed explicitly and expressively.

4.4 Selecting Resources According to a Competency Scale

The competency referencing of activities, resources and actors in a Learning Design can be used in many ways to support the design and delivery of learning environments.

At design time, they can help designers (or learners acting as their own designers) to:

- prepare a sequence of learning units that should increase progressively the mastery level of learners,
- identify learning resources (documents, tools, activities, persons) to be included in a learning design that possess the right knowledge and the right mastery level to support the learners' progress,

- provide criteria to form teams with learner having homogenous or heterogeneous mastery levels, or to plan different paths or plays for learners with weak or strong mastery patterns,

At delivery time, they can help learners and trainers to:

- evaluate the progress of learners' competencies for important knowledge elements,
- detect learners at risk by comparing their evolution pattern to the group average and alert learners, trainers and designers on possible flaws in the learning environment,
- find appropriate resources or units of learning in a learning object repository where resources have been referenced with semantic annotations,
- build /maintain user models to guide trainers' intervention, to trigger an intelligent advisor or a tutoring system, or to add information to an e-portfolio system.

Figure 14 illustrates the use of a bi-dimensional skill/performance scale. Here the mastery of a knowledge element, Multimedia Production Method, is evaluated. The skills scale, from 0 to 10, is complemented by a performance scale,

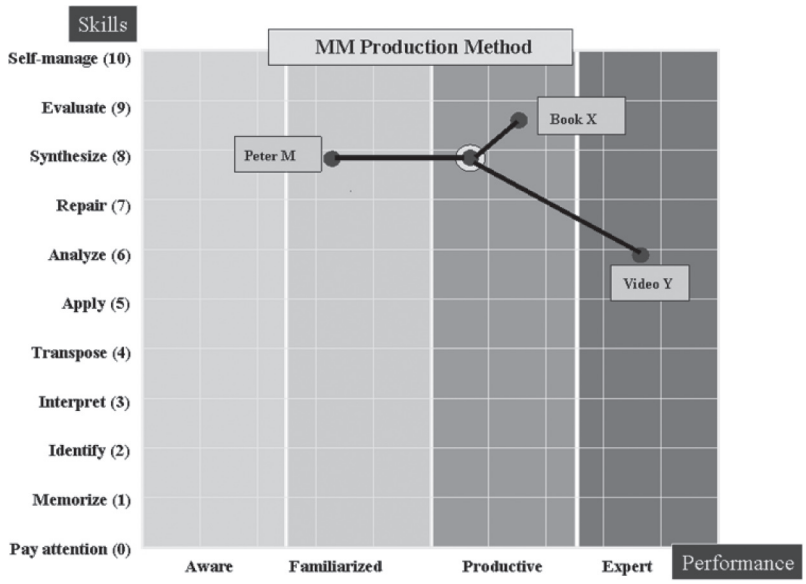


FIGURE 14
A Skill/Performance Scale for Resource Selection.

where values are decimals from 0 to 0.9, corresponding to qualitative terms like "A-Aware", "B-Familiarized", "C-Productive" or "D-Expert". On the figure, we see that PeterM's knowledge is evaluated at level 8.3 compared to a learning objectives at level 8.6: he can synthesize MM production methods, at a performance level showing he his familiarized with synthesizing such procedures. On the other hand, Book X is evaluated near 9.7; that might be too much for the actual competency of PeterM, unless we aim a target competency at that level or higher. On the other hand, a lecture in VideoY is evaluated near 6.9 so it is below PeterM's actual competency and might not prove very useful, except maybe as a review.

Another example of the use of competency scales is the concept of competency equilibrium around and activity where a learner interact with a tutor, using and producing resources, illustrated in figure 15.

This pattern illustrates part of a learning design where Act 5 is composed of four activities. Activity 5.1 and 5.2, preceding Activity 5.3, itself preceding (P link) Activity 5.4. Let us focus on activity 5.3, an activity performed by a learner and a trainer, both interacting with input resources A and B, helping the learner to produce a certain resource.

On this figure, we see that for a certain knowledge element in the ontology, for example "rice production processes", the target competency (TC) of activity 5.3

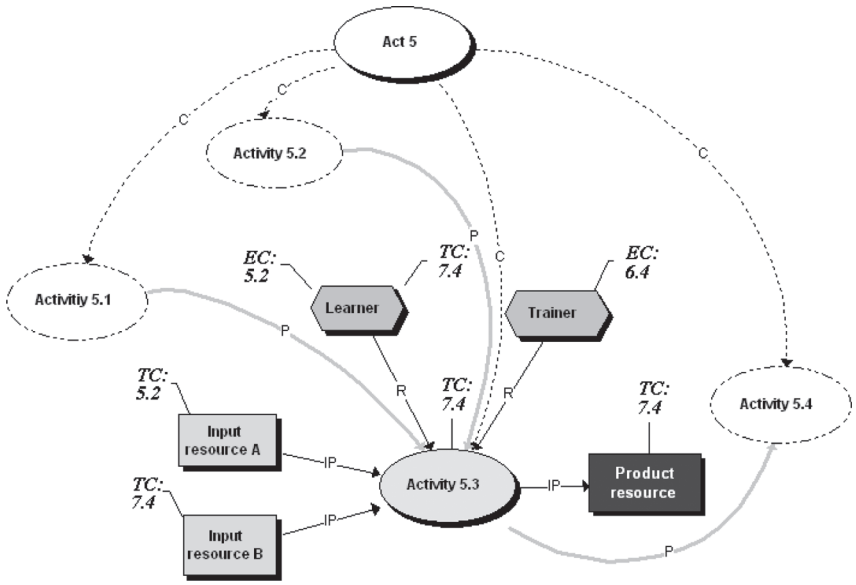


FIGURE 15
Associating knowledge mastery values to scenario components.

is evaluated at the 7.4 level on the skills/performance scale. (skill: "Repair", performance: B) so the product resource (an exam, an essay, a classification table,….) should show a TC level equal or higher than 7.4.

Since the learner has an entry competency (EC) of 5.2 (skill: "Apply", performance A), he needs help. Here we have a trainer with EC = 6.4 so he alone cannot bring the learner all the way up, but he can certainly help him fill part of the gap. Also, the learner can use two input resources. Resource A is at TC = 5.2 so it can only serve to test the entry competencies of the learner, to make sure he has the prerequisites. Fortunately, Resource B has a TC = 7.4, the right target, providing it is not a lecture that starts at 7.4, but maybe an aggregate of learning objects that can help him progress with the help of the trainer. By the way the trainer will also learn a little bit in the process, so at the end the activity, we could consider raising his EC for the next run of the activity.

There are many other situations to investigate where competency equilibrium situations such as these will prove useful, but this example shows that this kind of analysis, by humans, by machines or both, can bring more intelligence in learning environments before learning takes places (at design time), during learning (to help learners use available resources adequately) and after learning (to evaluate and improve designs).

5. USABILITY, INTEROPERABILITY AND GENERALITY ISSUES

A number of considerations have been taken into account while building the TELOS Scenario Editor, the Scenario Player, the Ontology Editor and the Task Manager.

5.1 Usability Issues

First of all, we aimed to *represent all levels of IMS-LD visually*, in order to simplify the design process, a result that is badly needed according to many authors and to our own experience in previous projects. Unlike the proposals in the IMS-LD specification, the conditions are not defined as a non ordered list accessible only at the method level. In our case, this would mean that they would all appear on the main graph of the scenario. This would blur the whole design that would look sometimes like a huge spaghetti of links and figures. More important, that would complicate the design process and the validation of an author's intended flow of control. The fact that the *conditions are inserted locally, where they are needed*, provides a clear picture of the author's intentions and simplifies the validation of the scenario flow. We let our IMS-LD export-to-LD parser gather

all the conditions and bundle them at the method level to comply with the specification while freeing author's from this burden.

Even at level A of the specification, the IDLD project mentioned earlier has pointed out a number of improvements that we have made to our initial MOT+LD visual editor.

- In the TELOS scenario editor, *the need for artificial graphic symbol for activity structures is completely eliminated*. Activity structures are modeled as functions and their sub-models can be a mix of other activity structures (also modeled as functions), activities and operations. Selections between activity structures are modeled using condition symbols and sequences of activities use precedence links.
- The designer will *not have to use the "Environment" symbol, except in limited cases.* These symbols make the graphs unnecessarily heavy and are a nuisance when making transformations to a scenario. In the TELOS Scenario Editor, the translator to IMS-LD creates environment symbols automatically by gathering all the resources having input-output links with an activity.
- The designer *doesn't need to introduce graphic item symbols carrying the concrete resource locations,* which makes graphical models a bit crowded. In the TELOS Scenario Editor, we give to the designer the freedom to associate a symbol to an instance in the ontology (which corresponds to an item or concrete resource) or to assign to it a class of resources to be instantiated at run time.
- In TELOS, *services are to be modeled as generic operations* where the resources involved in the operation (actors, documents and tools) are specified as an operation's port. To insert a service in a scenario, we only need to "unify" these ports to resources in the scenario. The detailed structure required by IMS-LD XML is automatically created by the translator to IMS-LD and added to the manifest file.
- In TELOS, *metadata referencing is replaced by semantic references* performed within the Scenario Editor by associating to a resource a class or an instance of an ontology, a much simpler operation for the designer. To reference resources with LOM or DC metadata, one way will be to include LOM or DC classes into the TELOS ontology; other ways will also have to be investigated.

5.2 Generality Issues

Even though TELOS scenarios and IMS Learning Design are both essentially multi-actor process models, the previous section underlines important differences. We cope with these differences by building graph parsers for import/export facilities. This architecture choice was done for the following reasons.

First, we wanted to be able to use the Scenario Editor as the main aggregation tool in TELOS. TELOS is a multi-layer system. Engineers have to aggregate existing software components (built possibly with different technologies) to create new ones, in order to extend the capabilities of the system. Technologists have to create or extend a Web platform by building scenarios for designers (instructional design methods) that includes a variety of design tools and documents (Paquette and Magnan, 2008). The constraints introduced in the method structure of IMS-LD do not take these use cases into account.

A second goal was to encompass business workflows as well as instructional scenarios for learners within the same Scenario Editor to enlarge the domains of application. For workflows, Business Process Model Notations, such as the BPMN specification, are more restricted than learning designs on certain aspects, but they provide a larger set of conditions to control the flow of activities. Unlike IMS-LD level B and C where all the conditions are declared at the method level, BPMN conditions are visually located at the point where they are used, thus given a more transparent view of the execution flow. Some of these features are also useful for learning designs, providing a larger base for the TELOS Scenario Editor.

6. CONCLUSION

There are limitations to the semantic referencing method we have presented here. Amongst others is the relation between actors and resources competencies in multi-actor processes. This is a research area that we need to explore further, in order to provide real help for the design of quality scenarios. Various resource equilibrium patterns will have to be investigated.

Another important stream of research is the support of emergent scenarios, such as those in project-based learning situations for examples or highly collaborative scenarios within the context of Web 2.0. In these situations, to start with, a very limited scenario is provided. After that, the scenario evolves by the learners themselves, achieving a design that is mainly theirs. Technically, our scenario editor actually is constructed at design time and executed as is, with of course some branching, but as planned by designers. To adapt it, the learners involved in a collaborative project-building activity, would have to come back in the designers environment to adapt the scenario. Ideally, this should be made possible within the run-time interface.

With regard to Educational Modelling, our goal was to provide a visual language that could be simple enough to support a wider use of specifications like

IMS-LD and, at the same time, be powerful enough to produce efficient and operational environments for the variety of situations encountered in learning and knowledge management. Such first results are encouraging, but we need more test-beds to improve the solution.

TELOS is a mature prototype, our next tasks will be to make it a robust industrial system[4] and implement it in a variety of applications, still working on increasing its user friendliness. We have the feeling that we have solved many of the scientific problems, but real-life applications have the ability to propose new and sometimes surprising research problems. Investigating such problems is a stimulating challenge that lies ahead.

REFERENCES

Baader et al, (2003) Baader, F.D. Calvanese, D.McGuinness, D. Nardi, P, Patel-Schneider, editors (2003) *The Description Logic Handbook*. Cambridge University Press.

Bloom (1975) Bloom B.S. *Taxonomy of Educational Objectives: the Classification of Educational Goals*. New York: D. McKay, 1975.

Breuker J. and Van de Velde W. (1994) *CommonKads Library for Expertise Modelling*. IOS Press, Amsterdam, 360 pages.

BPMN (2006) Business Processing Modelling Language http://www.bpmn.org/

Correal. D., Mariño O. (2007) Software *Requirements Specification Document for General Purpose Function's Editor (V0.4)*. LORNET Technical Documents, LICEF research centrer, Télé-université, Montreal.

Dalziel, J.R. (2005) LAMS. Learning Activity Management System 2.0. http://wiki.oamsfoundation.org/display/lams/Home.

Davies, J. Fensel, D, Van Harmelen, F. (2003) *Towards the Semantic Web, Ontology-Driven Knowledge Management*, Wiley, 288 pages.

Duval, E. & Robson, R. (2001) Guest Editorial on Metadata. *Interactive Learning Environments*, Special issue: Metadata, Volume 9–3, December 2001, pp. 201–206.

IDLD (2007) Implementation and Deployment of Learning Design – the Portal, www.idld.org.

IMS-RDCEO (2002). *IMS Reusable Definition of Competency or Educational Objective - XML Binding, Version (2002).* 1.0 Final Specification, IMS Global Learning Consortium, Inc. Revision: 25 October 2002.

IMS-LD (2003). *IMS Learning Design. Information Model, Best Practice and Implementation Guide, Binding document, Schemas*. Retrieved October 3, 2003, from http://www.imsglobal.org/learningdesign/index.cfm

Lundgren-Cayrol K., Marino O., Paquette G., Léonard M., de la Teja I. *Implementation and Deployment of IMS-LD - Outcomes of the IDLD Project*, ICALT-06 conference, Kerkrade, The Netherlands, June 2006.

Klein, M. and Fensel, D. (2001) *Ontology versioning for the semantic web*, International Semantic Web Working Symposium (SWWS), 2001.

[4] At the end of 2008, a three-year project has started with a large hydro-electricity company and the financial support of NSERC (Natural Science and Engineering Research Council of Canada), to build a commercial level version of TELOS supporting a knowledge management environment.

Koper R. (2002). *Modeling units of study from a pedagogical perspective – The pedagogical meta-model behind EML* Retrieved march 2002 from http://www.eml.ou.nl/introduction/articles.htm.

Magnan, F. and Paquette, G. (2006) *TELOS: An ontology driven eLearning OS*, SOA/AIS-06 Workshop, Dublin, Ireland, June 2006.

Mariño O., Casallas R., Villalobos J., Correal D., Contamines J. (2006) Bridging the Gap between e-learning Modeling and Delivery through the Transformation of Learnflows into Workflows. In Pierre, S., ed.: *Elearning networked. Environments and Architectures: A Knowledge Processing Perspective.* Springer Verlag.

Merrill M.D (1994). *Principles of Instructional Design.* Educational Technology Publications, Englewood Cliffs, New Jersey, 465 pages.

Minski M. (1975) A framework for reprsenting knowledge. In P. H. Winston (ED.), *The psychology of computer vision.* New York: McGraw-Hill.

Newell A & Simon H. (1972) *Human problem solving.* Englewood Cliffs, NF: Prentice-Hall.

Paquette G., Crevier, F., Aubin, C. (1994) ID Knowledge in a Course DesignWorkbench. *Educational Technology*, USA, volume 34, n. 9, pp. 50–57, November 1994.

Paquette G. (1996) La modélisation par objets typés: une méthode de représentation pour les systèmes d'apprentissage et d'aide a la tâche. *Sciences et techniques éducatives*, France, pp. 9–42, avril 1996.

Paquette, G. (1999) Meta-knowledge Representation for Learning Scenarios Engineering. Proceedings of AI-Ed'99 in AI and Education, open learning environments, S. Lajoie et M. Vivet (Eds), IOS Press, 1999.

Paquette G. (2001). *Designing Virtual Learning Centers.* In H. Adelsberger, B. Collis, J. Pawlowski (Eds) Handbook on Information Technologies for Education & Training within the Springer-Verlag series "International Handbook on Information Systems", (pp. 249–272).

Paquette G. (2002) TeleLearning Systems Engineering – Towards a new ISD model. *Journal of Structural Learning* 14, pp. 1–35.

Paquette G. (2003) *Instructional Engineering for Network-Based Learning.* Pfeiffer/Wiley Publishing Co, 262 pages.

Paquette, G. ,Léonard, M., Lundgren-Cayrol K., Mihaila S. and Gareau D. (2006) Learning Design based on Graphical Knowledge-Modeling , *Journal of Educational technology and Society* ET&S , Special issue on Learning Design, January 2006.

Paquette, G., Rosca, I., Mihaila, S., Masmoudi, A. (2006) Telos, a service-oriented framework to support learning and knowledge management. In Pierre, S., ed.: *E-Learning Networked Environments and Architectures: a Knowledge Processing Perspective.* Springer-Verlag.

Paquette G. (2007) An Ontology and a Software Framework for Competency Modeling and Management. *Educational Technology and Society*, Special Issue on "Advanced Technologies for Life-Long Learning", Volume 10, Issue 3, 2007 pp. 1–21PAPER.

Paquette, G. and Magnan F., (2008) An Executable Model for Virtual Campus Environments in H.H. Adelsberger, Kinshuk, J.M. Pawlowski and D. Sampson (Eds.), *International Handbook on Information Technologies for Education and Training*, 2nd Edition, Springer, Chapter 19, pp. 365–405, June 2008.

Pitrat J. (1991). *Métaconnaissance, avenir de l'Intelligence Artificielle*. Hermès, Paris, 1991.

RELOAD (2006) RELOAD Project. Retrieved February 4, 2006 from http://www.reload.ac.uk

Romiszowski A. J. (1981) *Designing Instructional Systems.* Kogan Page London/Nichols Publising, New York, 415 pages.

Schreiber G., Wielinga B., Breuker J. (1993) *KADS – A Principled Approach to Knowledge-based System Development*. San Diego : Academic Press. 457 p.

Tennyson, R. & Rasch, M. (1988) Linking cognitive learning theory to instructional prescriptions. *Instructional Science*, 17, pp. 369–385.

Tetlow, P., Pan, J., Oberle, D., Wallace, E., Uschold, M., Kendall, E. (2001): *Ontology driven architectures and potential uses of the semantic web in systems and softrware engineering*. http://www.w3.org/2001/sw/BestPractices/SE/ODA/051126/ (2001)

W3C (2004) Ontology Web Language (OWL) Overview Document, www.w3.org/TR/2004/REC-owl-features-20040210/

West C. K., Farmer J. A., Wolff P. (1991) M. *Instructional Design, Implications from Cognitive Science.* Allyn and Bacon, Boston, 271 pages.

Wiley D.A. (2002). *Connecting learning objects to Instructional design theory: a definition, a metaphor, and a taxonomy*. In Wiley (Ed) The Instructional Use of Learning Objects. Agency for Instructional Technology and Association for Educational Communications of Technology, Bloomington, Indiana, 281 pages.

Wilson, S., Blinco, K., Rehak, D. (2004): Service-oriented frameworks: Modelling the infrastructure for the next generation of e-learning systems. White Paper presented at alt-i-lab 2004.